COS20007 Object-Oriented Programming

Topic 05 Part B Exceptions

Learning Outcomes

- The importance of exception handling in OOP
- Understand how to implement exception using try/catch blocks

Libraries provide a wide range of useful abstractions

You can get a text representation to print to the Terminal
Will be used to represent a playing card
Use this template to create Card objects
a Card can do ...
a Card 'Knows' ...
Cards can be face up or face down

Class that do ...

For example:

Console.WriteLine("Hello")
Int32.Parse("2024")
SplashKit.Color

Developers create applications, building on the available class libraries

You can get a text representation to print to the Terminal
Will be used to represent a playing card
Use this template to create Card objects
a Card can do ...
a Card 'Knows' ...
Cards can be face up or face down

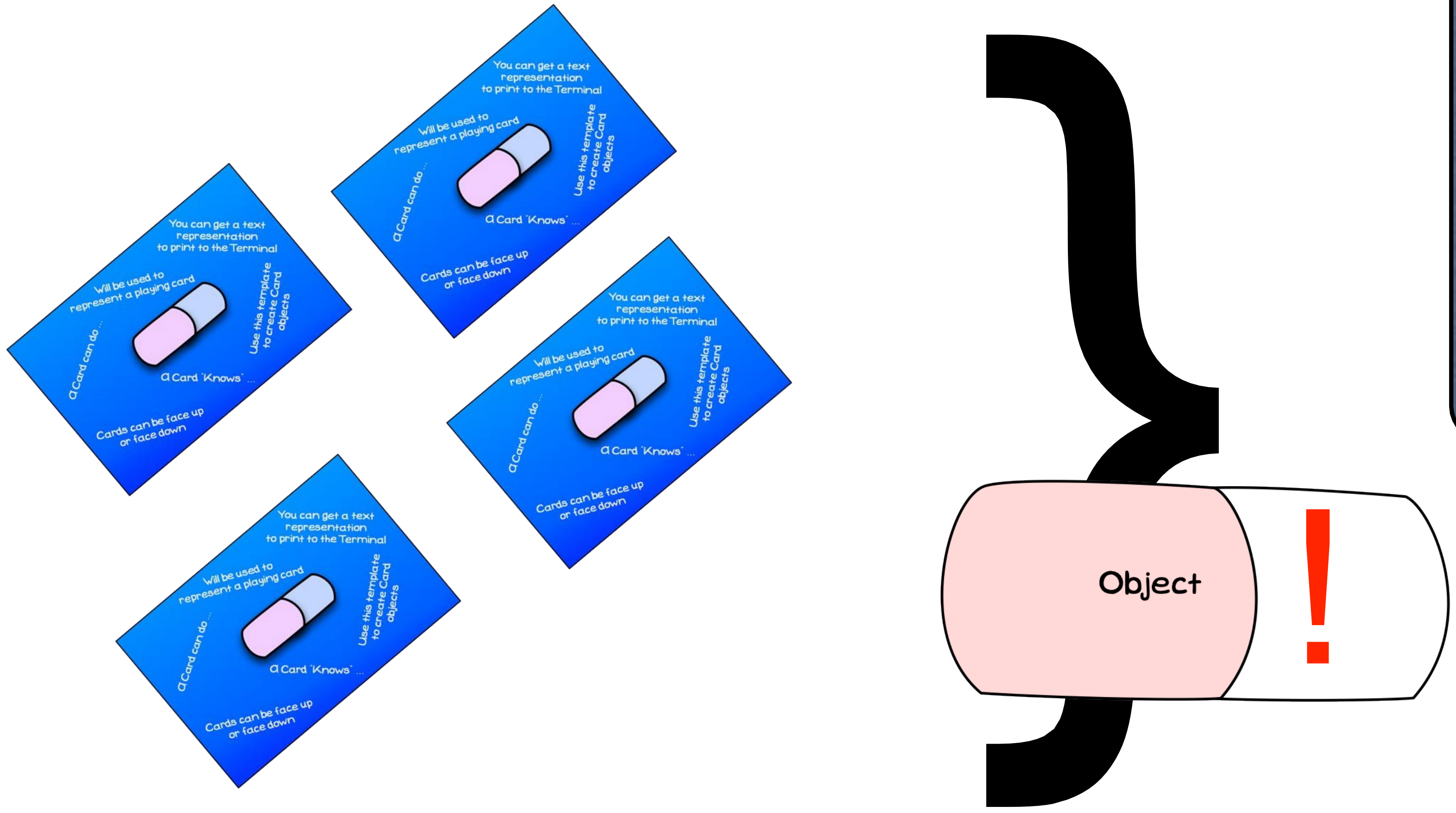
Use these classes to help you build...

Many libraries use exceptions to report errors they encounter

You can get a text representation to print to the Terminal
Will be used to represent a playing card
Use this template to create Card objects
a Card can do ...
a Card 'Knows' ...
Cards can be face up or face down
Object
!
That didn't work... argh... here handle this!

Overflow exception
Null Pointer exception
....

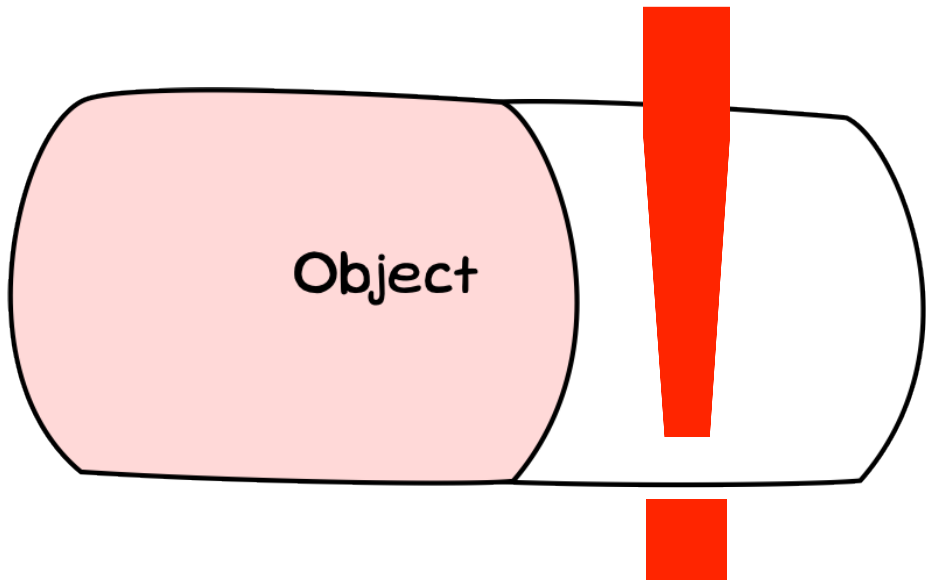
To use these libraries you need to learn how to handle these exceptions


You can get a text representation to print to the Terminal
Will be used to represent a playing card
Use this template to create Card objects
a Card can do ...
a Card 'Knows' ...
Cards can be face up or face down
Object
!

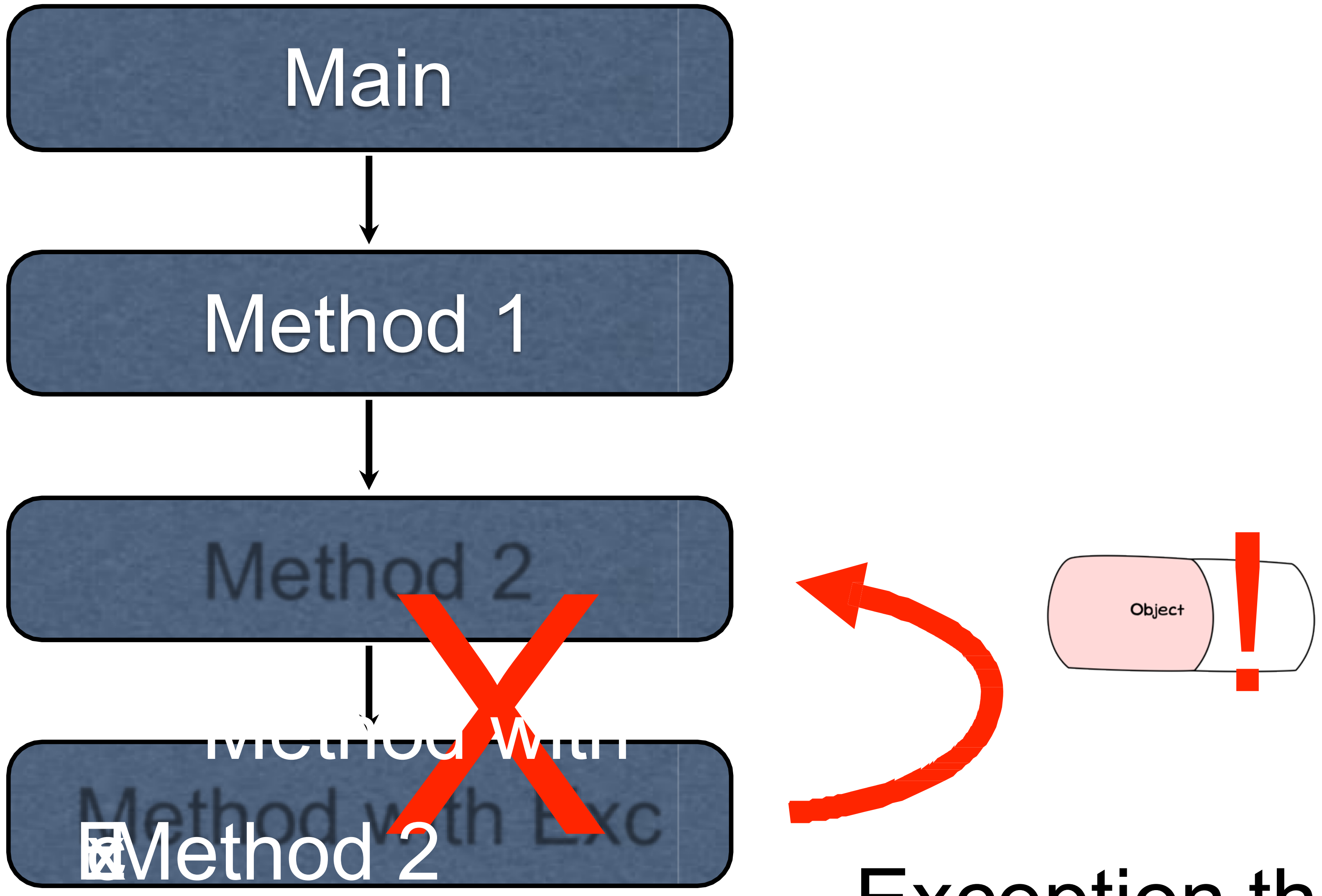
Calm down... I'm sure we can handle this...

Exceptions provide an alternate way of ending method calls

Exceptions are objects that contain an error message

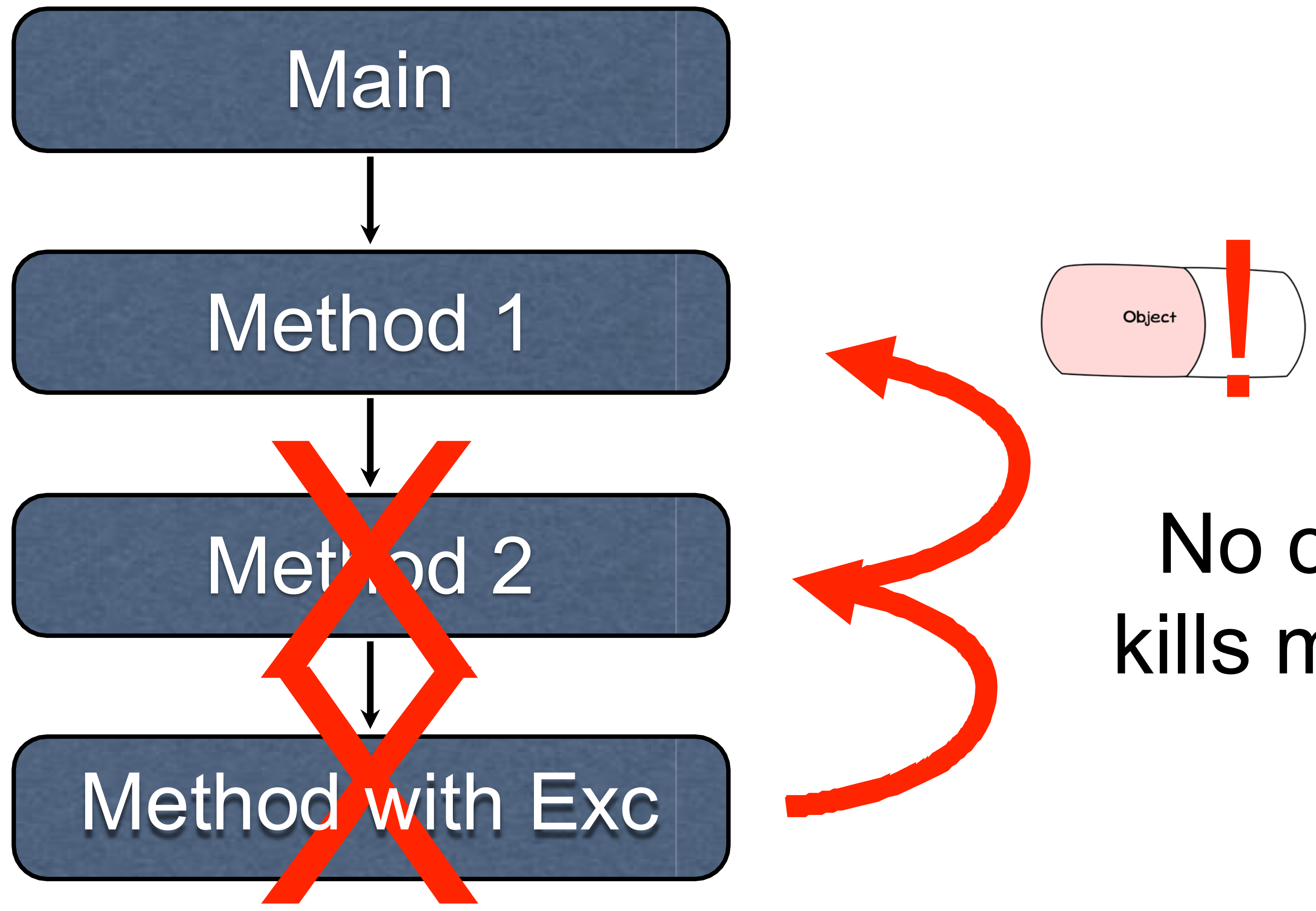


Throwing an exception causes methods to terminate until it is caught…



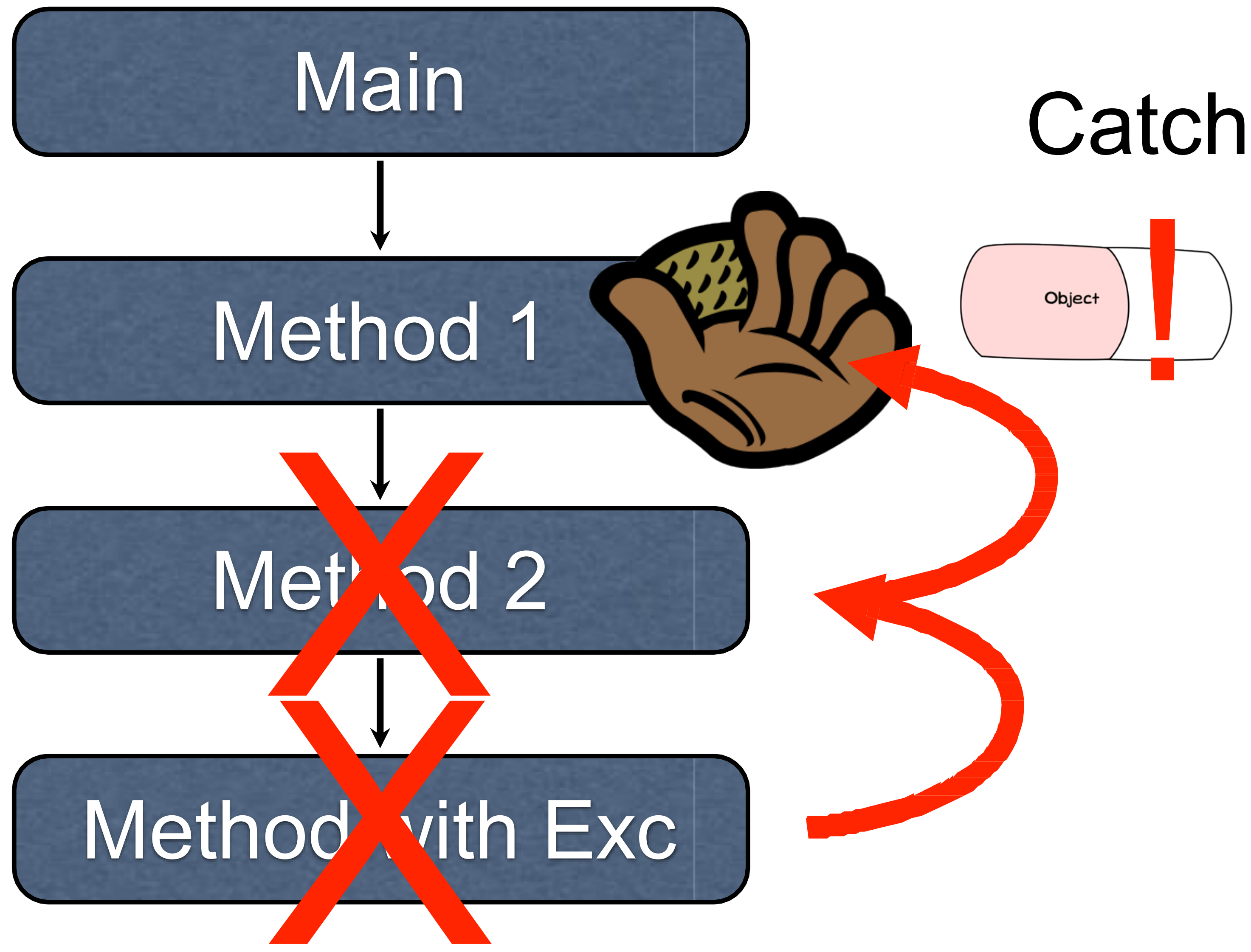
Exception thrown

Throwing an exception causes methods to terminate until it is caught…



No catch...
kills method 2

When dealing with exceptions, try to perform the code and catch any exceptions



Catch handles the exception

```
try
{
    ...
}
catch (System.Exception e)
{
    // cleanup
    ...
}
```

Avoid using exceptions for known error conditions

Try to fail gracefully, think of exceptions as a child having a tantrum

Exceptions are:

- slow
- Make code harder to follow
- Try/catch everywhere
- Terminated if not handled correctly

Only use exceptions in exceptional circumstances

Errors you handle

Exceptions are for things you didn't think of

SWINBURNE UNIVERSITY OF TECHNOLOGY

Watch out for exceptions thrown from libraries you use

You can get a text representation to print to the Terminal
Will be used to represent a playing card
Use this template to create Card objects
a Card can do ...
a Card 'Knows' ...
Cards can be face up or face down

}

Please try to ...

Learn to deal with exceptions

Make sure you catch all possible exceptions (in C#) ...

```
/// <exception cref="IllegalStateException">Why it's thrown.</exception>
public void Add()
{
    int v1, v2, result;

     if ( _operands.Length < 2 )
     {
         throw new IllegalStateException("Add requires at least 2 operands.");
     }

     v1 = pop();
     v2 = pop();
     result = v1 + v2;
     push(result);
}
```

may throw this exception

Use catch block to deal with the error

```
try
{
    ...
}
catch (System.Exception e)
{
    // cleanup
    ...
}
```

Ok... it threw an exception. I need to clean up this mess!

Example

```
string filePath = Console.ReadLine();
try
{
    StreamReader reader = new StreamReader(filePath)
     Console.WriteLine("File opened successfully.");
}
catch (FileNotFoundException ex)
{
     Console.WriteLine(”Error: File not found");
}
catch (IOException ex)
{
     Console.WriteLine(”Error while accessing file");
}
catch (System.Exception ex)
{
     Console.WriteLine(”Unexpected");
}
```

SWINBURNE UNIVERSITY OF TECHNOLOGY

Use finally blocks to perform code regardless of how things end up

If this starts...

```
try
{
    ...
}
catch (Exception e)
{
    // cleanup
    ...
}
finally
{
    ...
}
```

This will run when it ends... exception or not!

Take away message

- Exceptions are one way of reporting errors in the code
- Only use exceptions for exceptional cases that **we cannot anticipate**
- Learn to handle other's exceptions, and report errors gracefully yourself
- Exceptions: objects can have tantrums too
- Exception cases should be documented in user documents